MIT-ADT UNIVERSITY
PUNE, INDIA
A Leap Towards The World Class Education

School of Engineering & Sciences

Department of Electronics & Communication Engineering

LABORATORY MANUAL

Program:	**B. Tech.**		
Year:	**First Year**		
Course Code:	**23ECE1202**		
Course Name:	**Object Oriented Programming Laboratory**		
AY:	**2024-25**	**Sem:**	**Even**

CONTENTS

Exp. No	Name of Topic / Subtopic	Page No.
1.	Write a program in C++ to swap two numbers without using a third variable.	**2**
2.	Write a C++ program to print a multiplication table of a given number.	5
3.	Write a C++ program to find the length of a string without using standard function	9
4.	Write a program in C++ to sort the numbers in an array.	13
5.	Write a C++ program to read the data of N employees and compute the Net salary of each employee (DA=52% of Basic and Income Tax (IT) =30% of the gross salary).	18
6.	Given that an EMPLOYEE class contains the following members: data members: Employee number, Employee name, Basic, DA, IT, Net Salary and print data members.	23
7.	Write a C++ program create a calculator for an arithmetic operator (+ -, *, /). The program should take two operands from the user and perform the operation on those two operands depending upon the operator entered by the user.	27
8.	Write a program in C++ to perform following operations on complex numbers. 1)Add 2) Subtract 3) Multiply 4) Divide Use operator overloading for these operations.	32
9.	Write a program in C++ with class Rectangle with the data fields width, length, area and color. Create two objects of Rectangle and compare their area and color. If the area and color both are the same for the objects then display “Matching Rectangles”, otherwise displa “ Non-matching Rectangle”.	37
10.	Write a Python program to calculate the area of a triangle.	41
11.	Write a Python program to find the largest among three numbers.	46

12.	Write a Python program to print all prime numbers up to a number.	50
13.	Write a Python program to accept a number from the user and print digits of the number in a reverse order.	52
14.	Write a python program that accepts a string from the user and performs the following string operations- i. Calculate length of string ii. String reversal iii. Equality check of two strings	55
15.	Create class EMPLOYEE for storing details (Name, Designation, gender, Date of Joining and Salary). Define function members to compute a)total number of employees in an organization b) count of male and female employee c) Employee with salary more than 10,00 d) Employee with designation “Asst Manager”	58
16.	Write a program in python to perform following operations on complex numbers. 1)Add 2) Subtract 3) Multiply 4) Divide Use operator overloading for these operations.	61

EXPERIMENT NO: 01

TITLE: - Write a program in C++ to swap two numbers.

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO: -

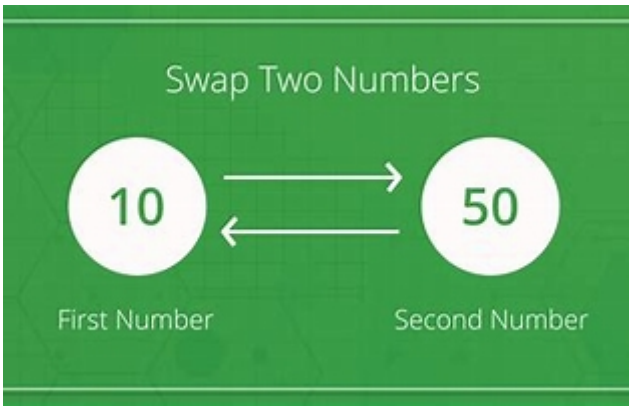
AIM: Write a program in C++ to swap two numbers without using third variable.

PROBLEM DEFINITION: Swapping of two numbers without using third variable

.

THEORY: Swapping Two Numbers Using Without Using Third Variable

In this variation of swapping two variables, we are not using any temporary variable to store the value. In the first variable we are storing the sum of both variable. Then, in next step we are we are extracting the value of 1st variable by subtracting the value of 2nd variable form the sum & storing it in 2nd variable. At last, we are extracting the original value of the 2nd variable & storing it in the 1st variable.

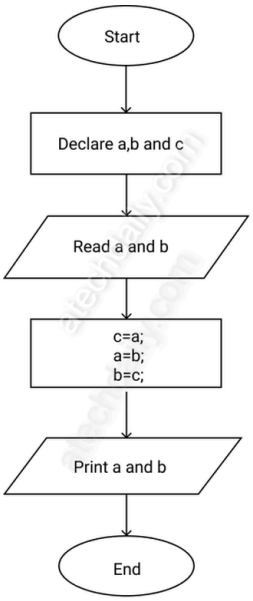


ALGORITHM:

Step 1: Declare 2 variables x and y.
Step 2: Read two numbers from the keyboard.
Step 3: Swap numbers.
//Apply addition and subtraction operations to swap the numbers.
i. x=x+y
ii. y=x-y
iii. x=x-y

Step 4: Print x and y values.

FLOWCHART:



PROGRAM CODE:

```
1.  #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int a=5, b=10;
6. cout<<"Before swap a= "<<a<<" b= "<<b<<endl;
7. a=a+b;
8. b=a-b;
9. a=a-b;
10. cout<<"After swap a= "<<a<<" b= "<<b<<endl;
11. return 0;
12. }
```

OUTPUT:

Before swap a= 5 b= 10
After swap a= 10 b= 5

CONCLUSION :

QUESTIONS:

1. . What are the Advantages of C++?

2. What are the different Data Types present in C++?

3. Define ‘std'

4. What is the difference between C and C++?

EXPERIMENT NO: 02

TITLE: - Write a C++ program to print a multiplication table of given numbers.

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL N0. :

AIM: Write a C++ program to print multiplication table of given number.

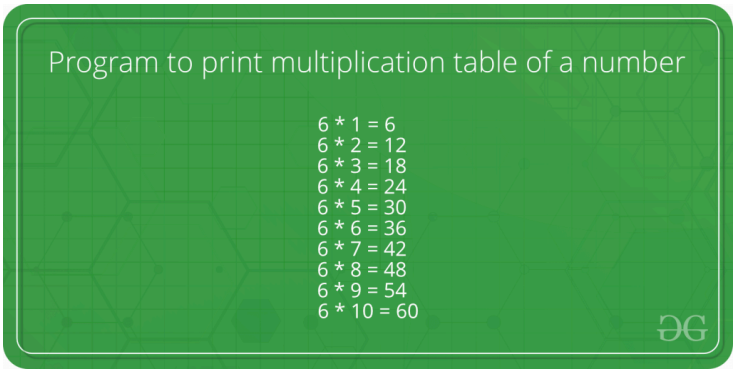
PROBLEM DEFINITION: to print multiplication table of given number

.

THEORY :

To print a table of any given number in C++ programming, you have to ask the user to enter the number. Then start multiplying that number from 1 to 10, one by one, and display the multiplication result at the time of multiplying, as shown here in the following program.

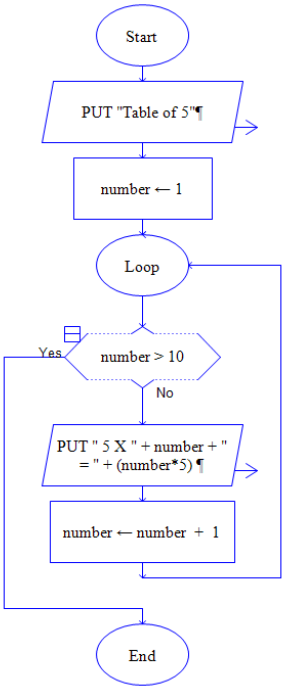
A multiplication table shows a list of multiples of a particular number, from 1 to 10. In this article, we will learn to generate and display the multiplication table of a number in C++ programming language.



ALGORITHM:

Step 1: Enter a number to print table at runtime.
Step 2: Read that number from keyboard.
Step 3: Using for loop print number*I 10 times.
// for(i=1; i<=10; i++)
Step 4: Print num*I 10 times where i=0 to 10.

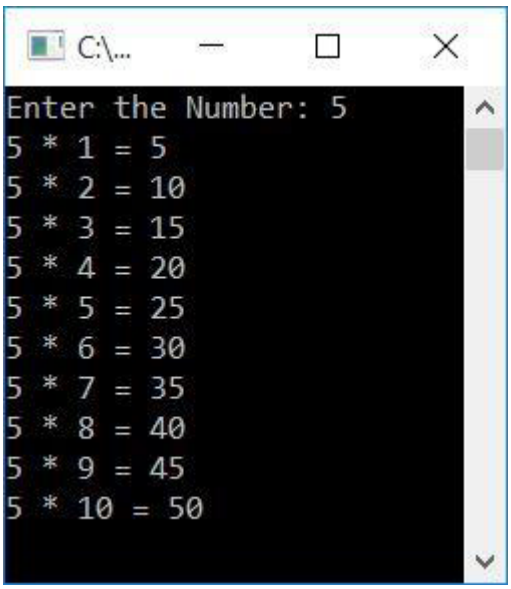
FLOWCHART:



PROGRAM CODE:

```
#include<iostream>
using namespace std;
int main()
{
int num, i, res;
cout<<"Enter the Number: ";
cin>>num;
for(i=1; i<=10; i++)
{
res = num*i;
cout<<num<<" * "<<i<<" = "<<res;
cout<<endl;
}
cout<<endl;
return 0;
}
```

OUTPUT:



CONCLUSION:

QUESTIONS:

1. What is the difference between a while loop and a do-while loop?

2. . Discuss the difference between prefix and postfix?

3. . Can you compile a program without the main function? Explain.

EXPERIMENT NO: 03

TITLE: - Charater Array(String)

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

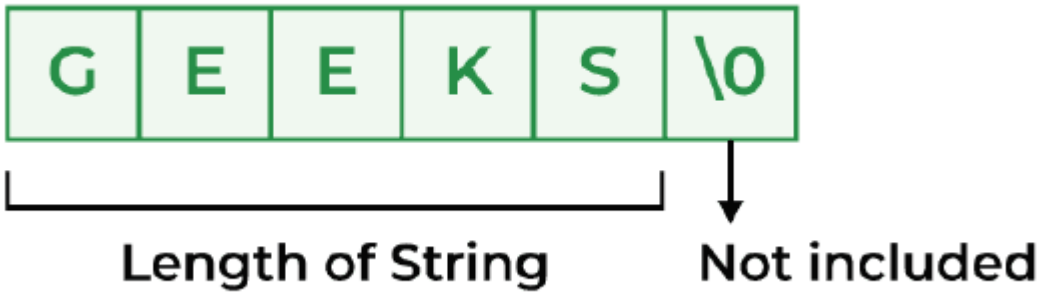
ROLL NO :

AIM: Write a C++ program to find the length of a string without using standard function.

PROBLEM DEFINITION: To find the length of a string without using standard function.

THEORY:

String is a sequence of characters. char data type is used to represent one single character in C++. So if you want to use a string in your program then you can use an array of characters. The declaration and definition of the string using an array of chars is similar to the declaration and definition of an array of any other data type. Any string ends with a terminating null character '\0'.

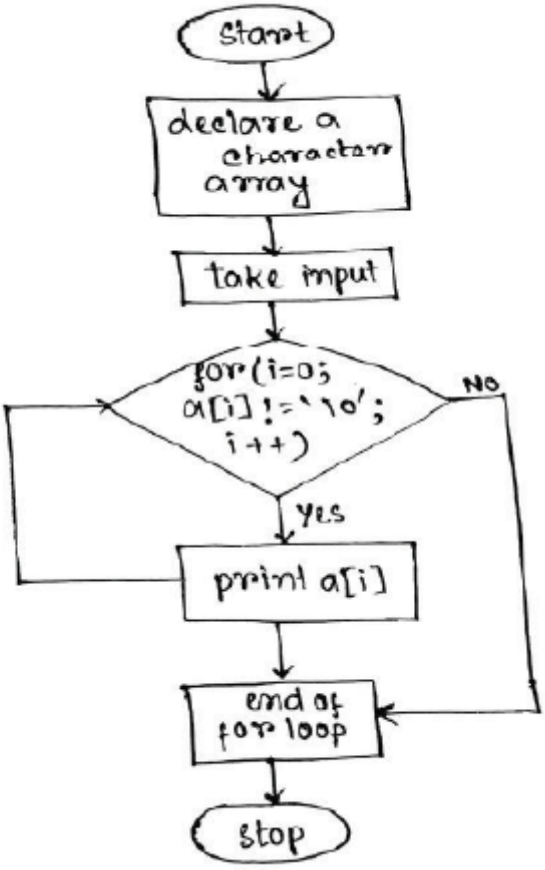


An array definition in such a way should include the null character '\0' as the last element. C++ provides the following two types of string representations: The C-style character string. The string class type introduced with Standard C++. The C-style character string originated within the C language and continues to be supported within C++. This string is actually a one-dimensional array of characters which is terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.

ALGORITHM:

1. Initialize a variable ‘str’ , ‘i’ , ‘length’.
2. Accept the value using ‘cin’ and ‘cout’.
3. Initiate a for loop.
4. Terminate the loop when null(‘\0’).
5. Print length.

FLOWCHART :



PROGRAM :

```
#include<iostream>
using namespace std;
int main()
{
int i,count=0; char ch[50];
```

```
cout<<"\nEnter any string :: ";
cin>>ch;
for(i=0;ch[i]!='\0';i++)
{
count++;
}
cout<<"\nLength of String [ "<<ch<<" ] is :: "<<count<<"\n";
return 0;
}
```

OUTPUT:

Enter any string :: Helloworld

Length of String is :: 10

CONCLUSION :

QUESTIONS:

1. What is a C++ string?

2. How do you declare and initialize a C++ string?

3. How can you access individual characters of a C++ string?

4. What is the difference between C-style strings and C++ strings?

EXPERIMENT NO: 04

TITLE: - Application of an Array.

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

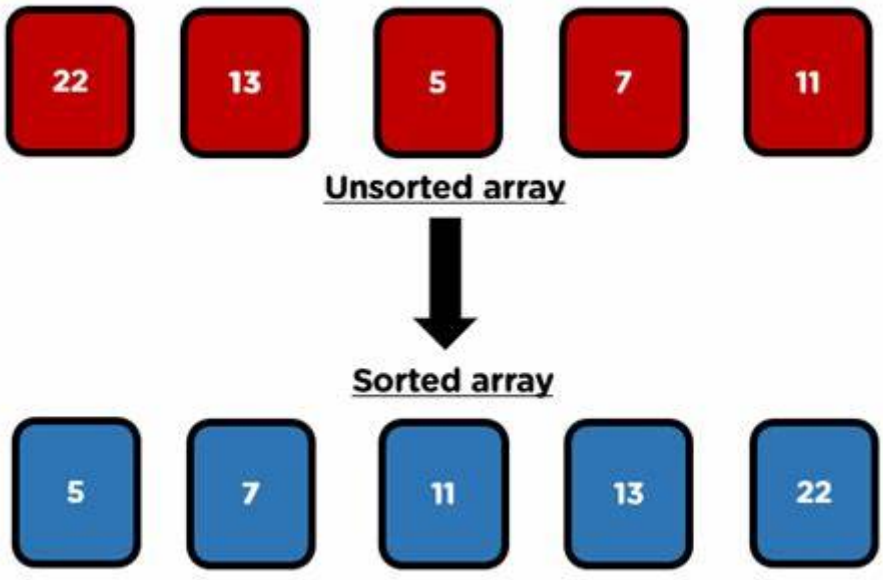
ROLL NO

AIM: Write a program in C++ to sort the numbers in an array.

PROBLEM DEFINITION:sorting the array

THEORY:

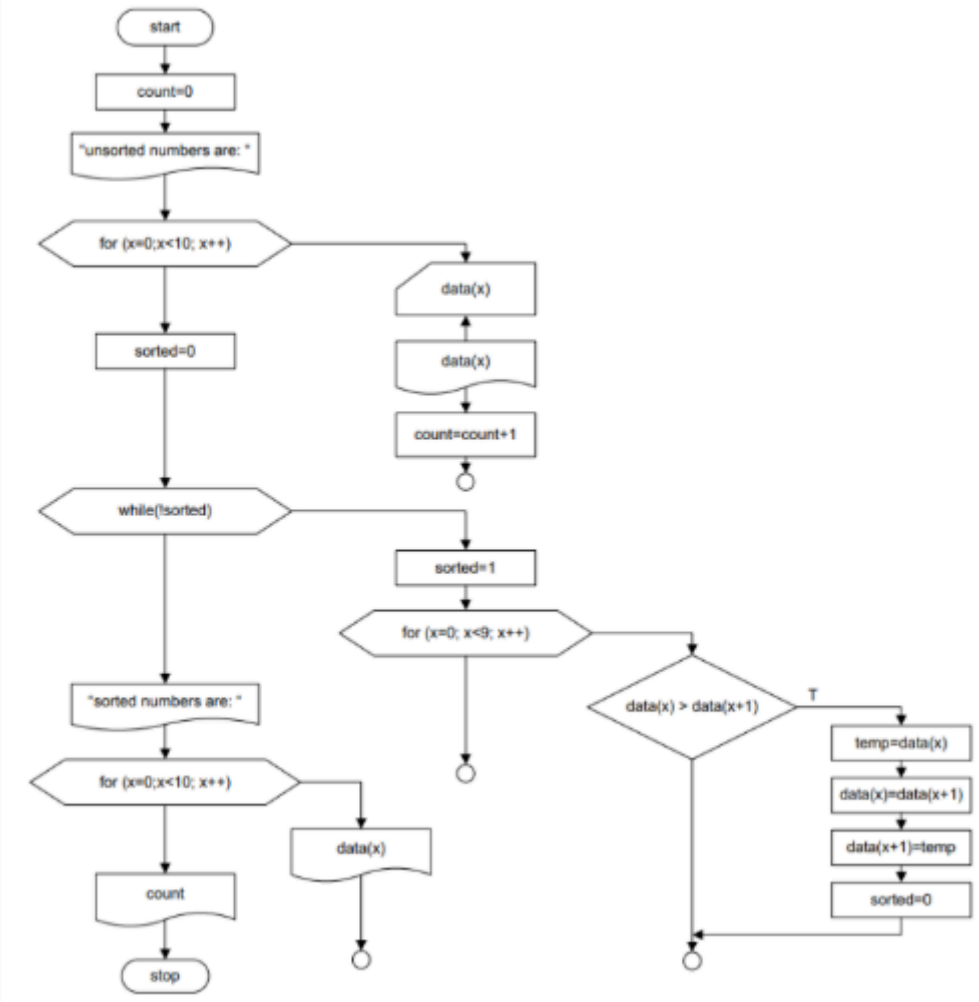
An Array is a collection of data of the same data type, stored at a contiguous memory location. Indexing of an array starts from 0. It means the first element is stored at the 0th index, the second at 1st, and so on. Elements of an array can be accessed using their indices. Once an array is declared its size remains constant throughout the program. An array can have multiple dimensions. The number of elements in an array can be determined using the sizeof operator. We can find the size of the type of elements stored in an array by subtracting adjacent addresses.



ALGORITHM:

1. Take the size of the array from the user.
2. Declare an array of given input size.
3. Take the input of all elements of the array.
4. Now run a for loop from 0 to size-1.
5. And for every element check it from all the next elements to it. If the element is greater than swap that number.
6. In this way the array will get sorted in ascending order.

FLOWCHART :



PROGRAM :

```
#include<iostream>
using namespace std;
int main()
{
   int i, j, size, temp; int arr[25];
   // Asking for input
   cout << "Enter the total no. of elements: "; cin >> size;
   // Enter the elements
   cout << "Enter the elements of the array: " << endl;
   for (i = 0; i < size; i++)
   {
      cin >> arr[i]; }
      // Sorting elements in ascending order
      for (i = 0; i < size; i++)
      {
         for (j = i; j < size; j++)
         {
            if (arr[i] > arr[j+1])
            {
               temp = arr[i];
               arr[i] = arr[j+1];
               arr[j+1] = temp;

            }
         }

      }
   // Displaying output
   cout << "Elements sorted in the ascending order are: " << endl;
   for (i = 1; i <= size; i++)
   {
      cout << arr[i] << endl;
   }
   return 0;

}
```

OUTPUT :

```
Enter the total no. of elements: 10
Enter the elements of the array:
34
56
21
46
12
3
9
25
6
89
Elements sorted in the ascending order are:
6
9
12
21
25
34
46
56
89
```

CONCLUSION:

QUESTIONS :

1. How do you declare an Array?

2. Mention some advantages and disadvantages of Arrays.

3. What will happen if you do not initialize an Array?

4. What is the default value of Array in Java?

EXPERIMENT NO: 05

TITLE: - Study of concept of class and object.

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO. :

AIM: Write a C++ program to read the data of N employees and compute the Net salary of each employee (DA=52% of Basic and Income Tax (IT) =30% of the gross salary). Given that an EMPLOYEE class contains the following members: data members: Employee number, Employee name, Basic, DA, IT, Net Salary and print data members.

PROBLEM DEFINITION: Classes and Object

THEORY:

1. Class:

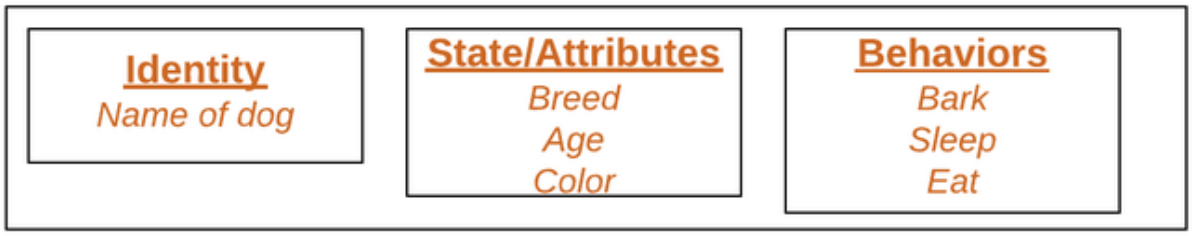
A class is a user-defined data type. It consists of data members and member functions, which can be accessed and used by creating an instance of that class. It represents the set of properties or methods that are common to all objects of one type. A class is like a blueprint for an object.

For Example: Consider the Class of Cars. There may be many cars with different names and brands but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range, etc. So here, cars are the class, and wheels, speed limits, mileage are their properties.

2. Object:

It is a basic unit of Object-Oriented Programming and represents the real-life entities. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. An object has an identity, state, and behavior. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

For example "Dog" is a real-life Object, which has some characteristics like color, Breed, Bark, Sleep, and Eats.

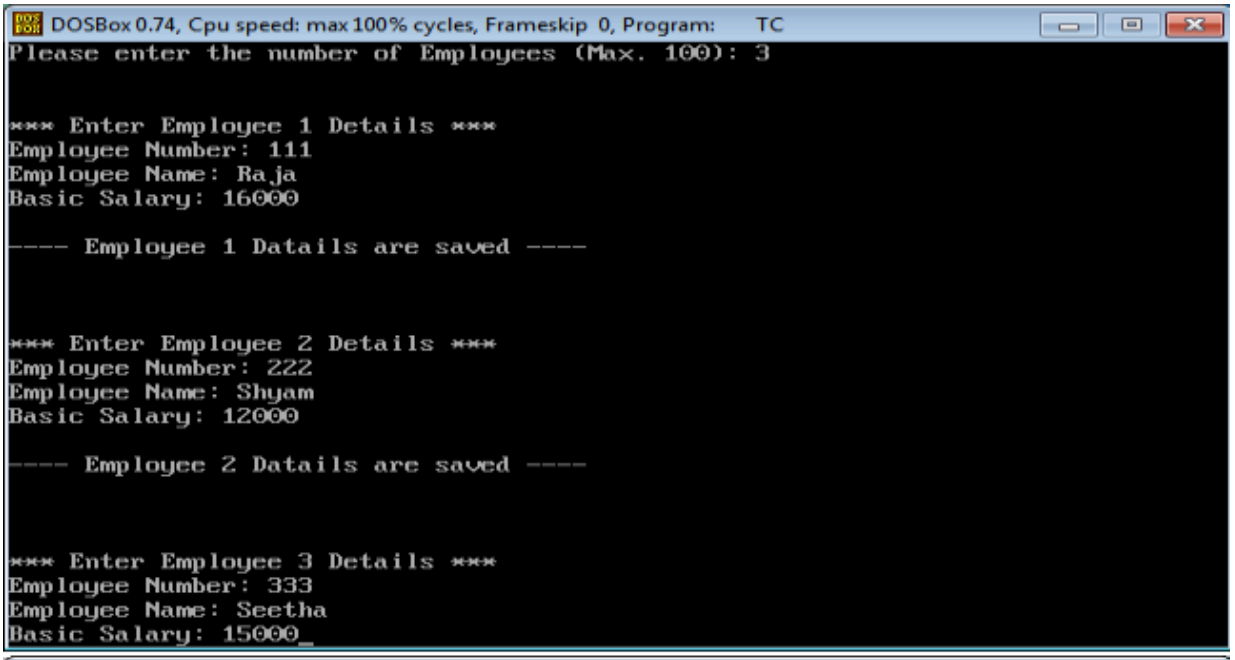


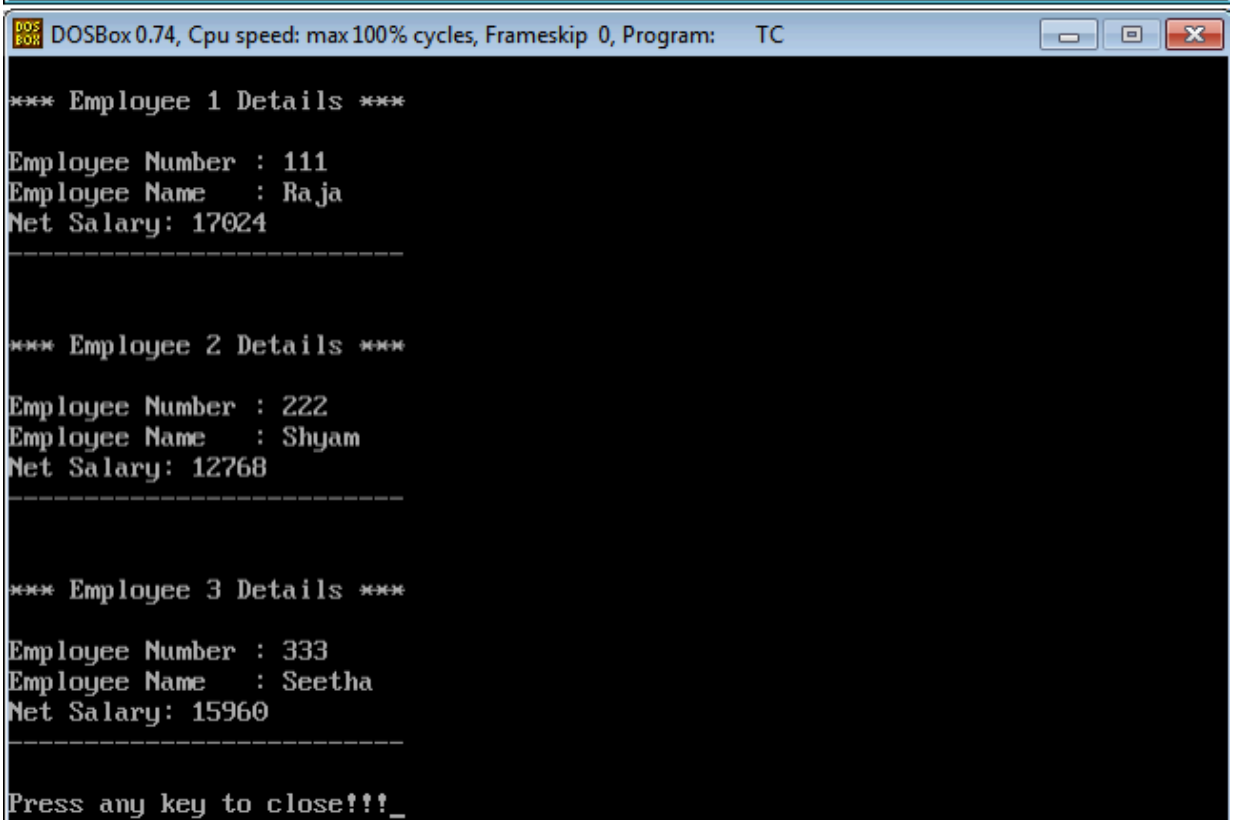
ALGORITHM:
Step 1 - Include the required header files (iostream.h, conio.h).
Step 2 - Create a class Employee with the following members. o emp_number, emp_name, basic, da, it, gross_salary, and net_salary as data members o read_emp_details(), find_net_salary(), and display_emp_details() as member functions.
Step 3 - Implement all the member functions with their respective code.
Step 4 - Create a main() method.
Step 5 - Create an array of class object with a specific size and number_of_emp as integer.
Step 6 - Read number_of_emp.
Step 7 - Call the read_emp_details() method through the array of class objects from 0 to number_of_emp.
Step 8 - Call the find_net_salary() method through the array of class objects from 0 to number_of_emp.
Step 9 - Call the display_emp_details() method through the array of class objects from 0 to number_of_emp.
Step 10 - return 0 to exit from the program execution.

PROGRAM :

```
#include<iostream>
using namespace std;
class Employee
{
   char emp_name[30];
   int emp_number;
   float basic, da, it, gross_salary, net_salary;
   public: void read_emp_details(int count)
   {
      cout<<"\n\n*** Enter Employee "<<count<<" Details ***";
      cout<<"\nEmployee Number: ";
      cin>>emp_number;
      cout<<"Employee Name: ";
      cin>>emp_name;
      cout<<"Basic Salary: ";
      cin>>basic;
      cout<<"\n---- Employee "<<count<<" Datails are saved ----\n\n";

   }
   float find_net_salary()
   {
```

```
        da = basic * 0.52;
        gross_salary = basic + da;
        it = gross_salary * 0.30;
        net_salary = (basic + da) - it;
        return net_salary;
    }
    void display_emp_details(int count)
    {
        cout<<"\n\n*** Employee "<<count<<" Details ***\n";
        cout<<"\nEmployee Number : "<<emp_number;
        cout<<"\nEmployee Name : "<<emp_name;
        cout<<"\nNet Salary: "<<net_salary;
        cout<<"\n-------------------------\n";

    }

};
int main()
{
    Employee emp[100]; int number_of_emp, count;

    cout<<"\nPlease enter the number of Employees (Max. 100): ";
    cin>>number_of_emp;
    for(count=0; count< number_of_emp; count++)
    {
        emp[count].read_emp_details(count+1);

    }
    for(count=0; count < number_of_emp; count++)
    {
        emp[count].find_net_salary();

    }
    for(count=0; count < number_of_emp; count++)
    {
        emp[count].display_emp_details(count+1);

    }
    cout<<"\nPress any key to close!!!";
    return 0;

}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Please enter the number of Employees (Max. 100): 3
*** Enter Employee 1 Details ***
Employee Number: 111
Employee Name: Raja
Basic Salary: 16000
---- Employee 1 Datails are saved ----
*** Enter Employee 2 Details ***
Employee Number: 222
Employee Name: Shyam
Basic Salary: 12000
---- Employee 2 Datails are saved ----
*** Enter Employee 3 Details ***
Employee Number: 333
Employee Name: Seetha
Basic Salary: 15000_


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
*** Employee 1 Details ***
Employee Number : 111
Employee Name : Raja
Net Salary: 17024
*** Employee 2 Details ***
Employee Number : 222
Employee Name : Shyam
Net Salary: 12768
*** Employee 3 Details ***
Employee Number : 333
Employee Name : Seetha
Net Salary: 15960
Press any key to close!!!_

CONCLUSION:

QUESTIONS :

1. What is Object Oriented Programming (OOPs)

2.What is Encapsulation?

3. What is Abstraction?

EXPERIMENT NO: 06

TITLE: - Study of concept of constructor

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO. :

AIM: Write a C++ program create a calculator for an arithmetic operator (+, -, *, /). The program should take two operands from the user and perform the operation on those two operands depending upon the operator entered by the user.

PROBLEM DEFINITION: Write a C++ program create a calculator for an arithmetic operator (+, -, *, /). The program should take two operands from the user and perform the operation on those two operands depending upon the operator entered by the user.

THEORY:

Constructor in C++ is a special method that is invoked automatically at the time an object of a class is created. It is used to initialize the data members of new objects generally. The constructor in C++ has the same name as the class or structure. It constructs the values i.e. provides data for the object which is why it is known as a constructor.

Syntax of Constructors in C++

The prototype of the constructor looks like this:

<class-name> (){

...

}

Characteristics of Constructors in C++

- The name of the constructor is the same as its class name.
- Constructors are mostly declared in the public section of the class though they can be declared in the private section of the class.
- Constructors do not return values; hence they do not have a return type.
- A constructor gets called automatically when we create the object of the class.
- Types of Constructor Definitions in C++

In C++, there are 2 methods by which a constructor can be declared:

1. Defining the Constructor Within the Class

```
<class-name> (list-of-parameters) {
    // constructor definition
}
```

2. Defining the Constructor Outside the Class

```
<class-name> {

   // Declaring the constructor
   // Definiton will be provided outside
   <class-name>();

   // Defining remaining class
}

<class-name>: :<class-name>(list-of-parameters) {
     // constructor definition
}
```

PROGRAM :

```
#include <iostream>

using namespace std;

class Calculator {
private:
   double num1, num2;
   char op;

public:
   // Constructor to take input and perform the operation
   Calculator(double a, double b, char operation) {
      num1 = a;
      num2 = b;
      op = operation;

      // Perform the operation based on user input
      switch (op) {
         case '+':
            cout << "Result: " << num1 << " + " << num2 << " = " << (num1 + num2) << endl;
            break;
         case '-':
            cout << "Result: " << num1 << " - " << num2 << " = " << (num1 - num2) << endl;
            break;
         case '*':
            cout << "Result: " << num1 << " * " << num2 << " = " << (num1 * num2) << endl;
```

```
            break;
        case '/':
            if (num2 != 0)
                cout << "Result: " << num1 << " / " << num2 << " = " << (num1 / num2) << endl;
            else
                cout << "Error: Division by zero is not allowed." << endl;
            break;
        default:
            cout << "Error: Invalid operator! Please enter +, -, *, or /." << endl;
    }
  }
};

int main() {
  double a, b;
  char operation;

  // Take user input
  cout << "Enter first number: ";
  cin >> a;
  cout << "Enter an operator (+, -, *, /): ";
  cin >> operation;
  cout << "Enter second number: ";
  cin >> b;

  // Create an object of Calculator and perform the operation
  Calculator calc(a, b, operation);

  return 0;
}
```

OUTPUT :

```
Enter first number: 34
Enter an operator (+, -, *, /): *
Enter second number: 6
Result: 34 * 6 = 204
```

CONCLUSION:

QUESTIONS :

1. . What is the use of a constructor?

2. What if I don't use a copy constructor? Where does it create a problem?

3. Does C++ compiler create a default constructor when we write our own?

EXPERIMENT NO: 07

TITLE: - Study of concept of Operator Overloading

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO. :

AIM: Write a program in C++ to overload the unary operator.

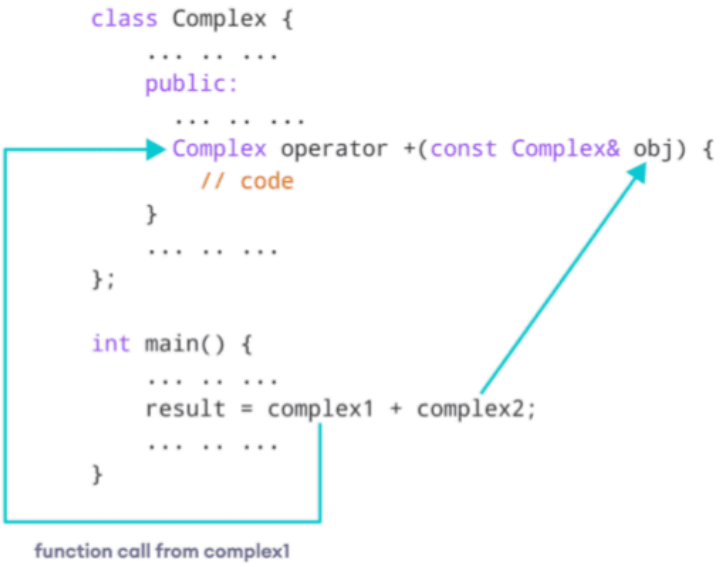
PROBLEM DEFINITION: Write a program in C++ to overload the unary operator.

THEORY:

C++ Operator Overloading
In C++, we can define how operators behave for user-defined types like class and structures. For example,

The + operator, when used with values of type int, returns their sum. However, when used with objects of a user-defined type, it is an error. In this case, we can define the behavior of the + operator to work with objects as well.

This concept of defining operators to work with objects and structure variables is known as operator overloading.



Syntax for C++ Operator Overloading
The syntax for overloading an operator is similar to that of function with the addition of the operator keyword followed by the operator symbol.

```
returnType operator symbol (arguments) {
    ... .. ...
}
```
Here,

returnType - the return type of the function
operator - a special keyword
symbol - the operator we want to overload (+, <, -, ++, etc.)
arguments - the arguments passed to the function

PROGRAM :

```
#include <iostream>
using namespace std;

class Number {
private:
    int value;

public:
    // Constructor
    Number(int v = 0) {
        value = v;
    }
```

```
    // Overloading the unary '-' operator
    Number operator-() {
        return Number(-value);
    }

    // Overloading prefix '++' operator
    Number operator++() {
        ++value;
        return *this;
    }

    // Overloading postfix '++' operator
    Number operator++(int) {
        Number temp = *this;
        value++;
        return temp;
    }

    // Overloading prefix '--' operator
    Number operator--() {
        --value;
        return *this;
    }

    // Overloading postfix '--' operator
    Number operator--(int) {
        Number temp = *this;
        value--;
        return temp;
    }

    // Function to display the number
    void display() {
        cout << "Value: " << value << endl;
    }
};

int main() {
    // Creating an object with an initial value
    Number num(5);

    cout << "Original number: ";
    num.display();

    // Applying unary minus operator
```

```
    Number negNum = -num;
    cout << "After applying unary '-': ";
    negNum.display();

    // Applying prefix increment
    ++num;
    cout << "After prefix '++': ";
    num.display();

    // Applying postfix increment
    Number temp1 = num++;
    cout << "After postfix '++' (before increment): ";
    temp1.display();
    cout << "After postfix '++' (after increment): ";
    num.display();

    // Applying prefix decrement
    --num;
    cout << "After prefix '--': ";
    num.display();

    // Applying postfix decrement
    Number temp2 = num--;
    cout << "After postfix '--' (before decrement): ";
    temp2.display();
    cout << "After postfix '--' (after decrement): ";
    num.display();

    return 0;
}
```

OUTPUT :

```
Original number: Value: 5
After applying unary '-': Value: -5
After prefix '++': Value: 6
After postfix '++' (before increment): Value: 6
After postfix '++' (after increment): Value: 7
After prefix '--': Value: 6
After postfix '--' (before decrement): Value: 6
After postfix '--' (after decrement): Value: 5
```

CONCLUSION:

QUESTIONS :

1. What is function overloading and operator overloading? - C++

2. What is an overloading unary operator? - C++

3. Which is the correct example of a unary operator?

EXPERIMENT NO: 08

TITLE: -Study of concept of Operator Overloading

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO. :

AIM: Write a program in C++ to perform following operations on complex numbers.
1)Add 2) Subtract 3) Multiply 4) Divide. Use operator overloading for these operations.

PROBLEM DEFINITION :
Write a program in C++ to perform following operations on complex numbers. 1)Add 2) Subtract 3) Multiply 4) Divide Use operator overloading for these operations.

THEORY :

Operator overloading is a type of polymorphism in which a single operator is overloaded to give a user-defined meaning. Operator overloading provides a flexible option for creating new definitions of C++ operators.

Syntax

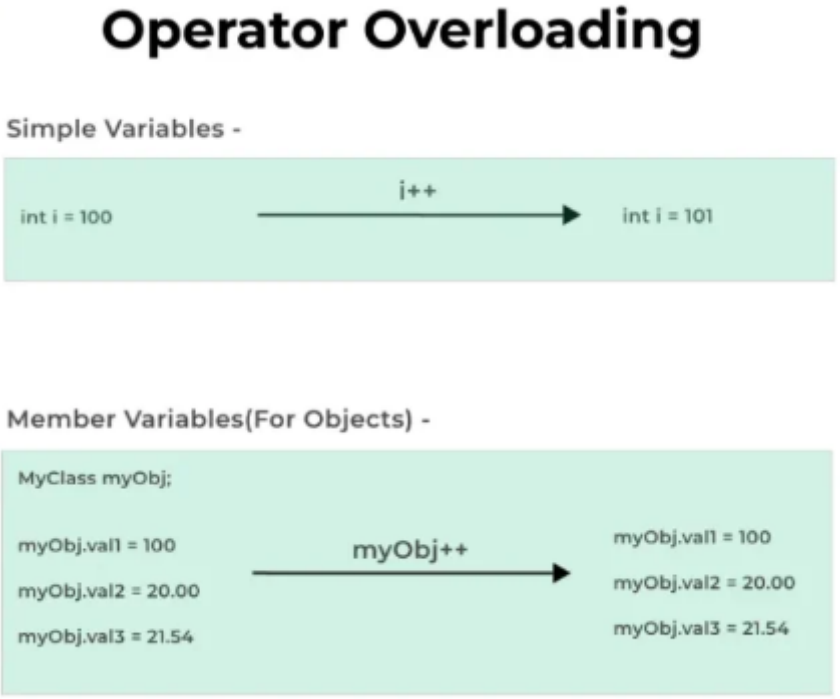
To overload a C++ operator, you should define a special function inside the Class as follows:

```
class class_name
{
   ... .. ...
   public
     return_type operator symbol (argument(s))
     {
        ... .. ...
     }
   ... .. ...
};
```

Here is an explanation for the above syntax:

- The return_type is the return type for the function.
- Next, you mention the operator keyword.
- The symbol denotes the operator symbol to be overloaded. For example, +, -, <, ++.

- The argument(s) can be passed to the operator function in the same way as functions.



PROGRAM :

```
#include <iostream>

using namespace std;

class Complex {
private:
   double real, imag;

public:
   // Constructor
   Complex(double r = 0, double i = 0) {
      real = r;
      imag = i;
   }

   // Overloading the + operator for addition
   Complex operator+(const Complex &c) {
      return Complex(real + c.real, imag + c.imag);
   }

   // Overloading the - operator for subtraction
   Complex operator-(const Complex &c) {
      return Complex(real - c.real, imag - c.imag);
   }
```

```
    // Overloading the * operator for multiplication
    Complex operator*(const Complex &c) {
        return Complex(real * c.real - imag * c.imag, real * c.imag + imag * c.real);
    }

    // Overloading the / operator for division
    Complex operator/(const Complex &c) {
        double denominator = c.real * c.real + c.imag * c.imag;
        if (denominator == 0) {
            cout << "Error: Division by zero is not allowed." << endl;
            return Complex(0, 0);
        }
        double realPart = (real * c.real + imag * c.imag) / denominator;
        double imagPart = (imag * c.real - real * c.imag) / denominator;
        return Complex(realPart, imagPart);
    }

    // Function to display the complex number
    void display() {
        cout << real;
        if (imag >= 0)
            cout << " + " << imag << "i" << endl;
        else
            cout << " - " << -imag << "i" << endl;
    }
};

int main() {
    double r1, i1, r2, i2;

    // Taking input for the first complex number
    cout << "Enter real and imaginary part of first complex number: ";
    cin >> r1 >> i1;

    // Taking input for the second complex number
    cout << "Enter real and imaginary part of second complex number: ";
    cin >> r2 >> i2;

    // Creating complex number objects
    Complex c1(r1, i1), c2(r2, i2), result;

    // Performing operations using overloaded operators
    cout << "\nResults:" << endl;

    result = c1 + c2;
    cout << "Addition: ";
    result.display();

    result = c1 - c2;
```

```
    cout << "Subtraction: ";
    result.display();

    result = c1 * c2;
    cout << "Multiplication: ";
    result.display();

    result = c1 / c2;
    cout << "Division: ";
    result.display();

    return 0;
}
```

OUTPUT :

```
Enter real and imaginary part of first complex number: 3 4
Enter real and imaginary part of second complex number: 6 2

Results:
Addition: 9 + 6i
Subtraction: -3 + 2i
Multiplication: 10 + 30i
Division: 0.65 + 0.45i
```

CONCLUSION :

QUESTIONS :

1. What is Polymorphism?

2. What are the rules for operator overloading?

3. What are the types of operator overloading?

4. What is the syntax for operator overloading?

EXPERIMENT NO: 09

TITLE: -Study of concept of Inheritance.

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO. :

AIM: Write a program in C++ with class Rectangle with the data fields width, length, area and colour. Create two objects of Rectangle and compare their area and colour. If the area and colour both are the same for the objects then display " Matching Rectangles", otherwise display " Non-matching Rectangle".

PROBLEM DEFINITION : To understand the concept of inheritance

THEORY :
The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object Oriented Programming in C++. In this article, we will learn about inheritance in C++, its modes and types along with the information about how it affects different properties of the class.

Syntax of Inheritance in C++

```
class  derived_class_name : access-specifier  base_class_name
{
    //    body ....
};
```

where,

class: keyword to create a new class
derived_class_name: name of the new class, which will inherit the base class
access-specifier: Specifies the access mode which can be either of private, public or protected. If neither is specified, private is taken as default.
base-class-name: name of the base class.

ALGORITHM:
1. Define the Rectangle class: Declare private data fields: width, length, area, and color. Implement a constructor to initialize the data fields. Implement accessor methods to retrieve the values of the data fields. Implement a method to calculate the area of the rectangle.
2. Create objects of the Rectangle class: Instantiate two Rectangle objects with different dimensions and colors.

3. Compare the objects: Retrieve the area and color of each object using accessor methods. Compare the areas and colors of the two objects. If both the areas and colors are the same, display "Matching Rectangles"; otherwise, display "Non-matching Rectangles"

PROGRAM :

```
#include <iostream>
#include <string>

using namespace std;

class Rectangle {
private:
    double width, length, area;
    string color;

public:
    // Constructor to initialize the rectangle
    Rectangle(double w, double l, string c) {
        width = w;
        length = l;
        color = c;
        area = width * length; // Calculating area
    }

    // Function to get the area
    double getArea() {
        return area;
    }

    // Function to get the color
    string getColor() {
        return color;
    }

    // Function to compare two rectangles
    bool isMatching(Rectangle r) {
        return (area == r.getArea() && color == r.getColor());
    }

    // Function to display rectangle details
    void display() {
        cout << "Rectangle [Width: " << width << ", Length: " << length
             << ", Area: " << area << ", Color: " << color << "]" << endl;
    }
};

int main() {
    double w1, l1, w2, l2;
    string c1, c2;
```

```
    // Taking input for first rectangle
    cout << "Enter width, length, and color of first rectangle: ";
    cin >> w1 >> l1;
    cin.ignore(); // To handle newline before color input
    getline(cin, c1);

    // Taking input for second rectangle
    cout << "Enter width, length, and color of second rectangle: ";
    cin >> w2 >> l2;
    cin.ignore();
    getline(cin, c2);

    // Creating rectangle objects
    Rectangle rect1(w1, l1, c1);
    Rectangle rect2(w2, l2, c2);

    // Display rectangle details
    cout << "\nDetails of Rectangle 1:" << endl;
    rect1.display();

    cout << "\nDetails of Rectangle 2:" << endl;
    rect2.display();

    // Comparing rectangles
    if (rect1.isMatching(rect2))
        cout << "\nMatching Rectangles" << endl;
    else
        cout << "\nNon-matching Rectangles" << endl;

    return 0;
}
```

OUTPUT :

```
Enter width, length, and color of first rectangle: 20 40 blue
Enter width, length, and color of second rectangle: 12 24 red

Details of Rectangle 1:
Rectangle [Width: 20, Length: 40, Area: 800, Color: blue]

Details of Rectangle 2:
Rectangle [Width: 12, Length: 24, Area: 288, Color: red]

Non-matching Rectangles
```

CONCLUSION :

QUESTIONS :

1. **How is inheritance implemented in C++?**

2. **Explain single inheritance with an example.**

3. **Explain multiple inheritance with an example.**

EXPERIMENT NO: 10

TITLE: -Study of basic concept of python

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO. :

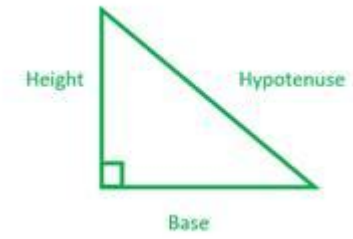
Aim : Write a Python program to calculate the area of a triangle.

Theory :

Area of a Right-Angled Triangle

In this example, we calculate the area of a right angled triangle whose base and height is given. First, we define a function to calculate the area where we write the formula to calculate the area and then return the answer. In the driver function we pass the values of base and height of the triangle.

Formula :



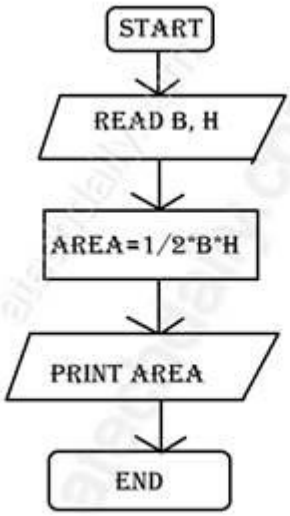
Area of Triangle: 1/2 * b * h

Algorithm

This algorithm is used to find the area of a right angle triangle. To Calculate area, we use only the base and height of a triangle.

1. Start
2. Input base, height
3. Calculate area = 1/2*base*height
4. print "Area of Triangle=" area
5. End

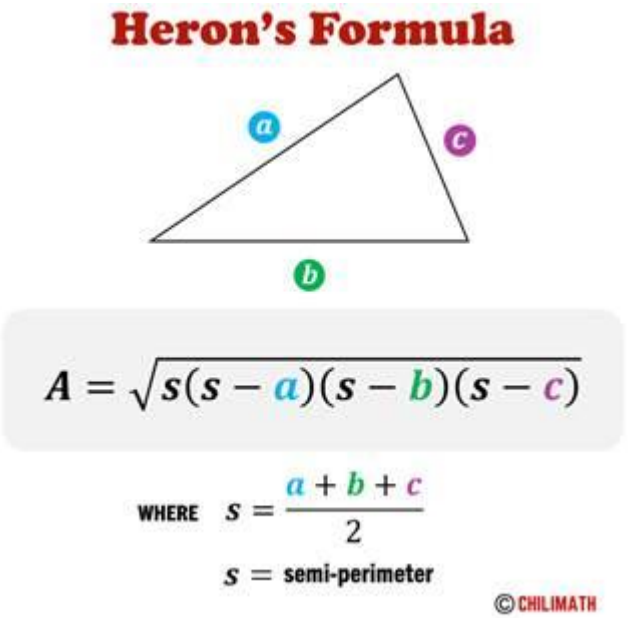
Flowchart :



Area of any Triangle in Python

In this example, we define a function for calculating the area of triangle at first where we use the formula and return the answer. In the driver function we pass the values as sides of triangle and call the function for calculating the area.

Heron's Formula

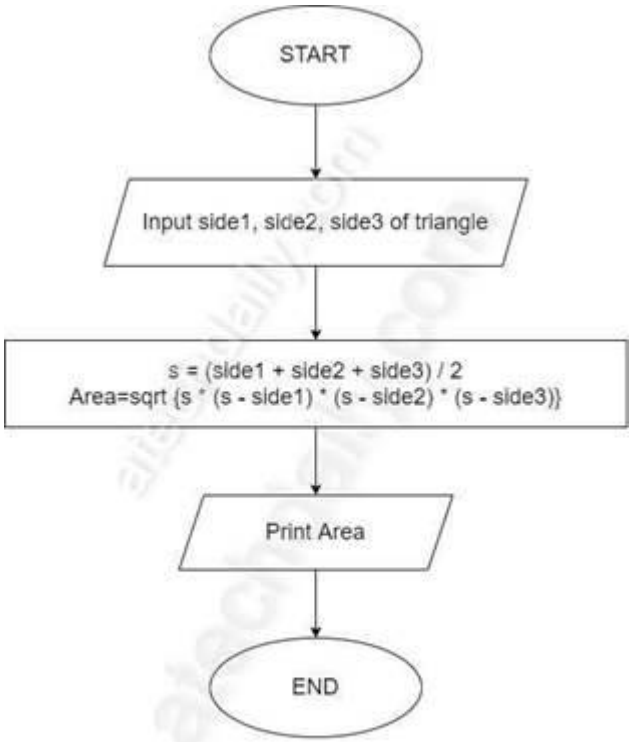


Area of Triangle = (s(s-a)*(s-b)*(s-c)) 0.5**

s = a + b + c/2

Algorithm:
This algorithm is used to find area of any triangle. To Calculate area, we use dimensions of all three sides of the triangle.
1. Start
2. Input side1, side2, side3
3. Calculate s = (side1 + side2 + side3) / 2
4. Calculate area = sqrt {s * (s - side1) * (s - side2) * (s - side3)}
5. Print "Area of Triangle=" area
6. End

Flowchart :



Program 1 :

```
def area_Triangle(b, h):
        area = 0.5 * b * h
        return area
```

```
base = 8

height = 9

area = areaTriangle(base, height)

print('Area = {:.2f}' .format(area))
```

OUTPUT :

```
Area = 36.00
```

Program 2 :

```
def area_Triangle(a, b, c):
        s = (a+b+c)/2
        area = (s*(s-a)*(s-b)*(s-c))**0.5
        return area


a = 7
b = 8
c = 9
aeron_area = area_Triangle(a, b, c)
print('Area = {:.2f}'.format(heron_area))
```

OUTPUT :

```
Area = 26.83
```

CONCLUSION :

Questions :

1. Define Python Language.

2. What are the Advantages of Python

2. What are the different Data Types present in Python?

EXPERIMENT NO: 11

TITLE: -Study of concept of if else in python

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO. :

Aim : Write a Python program to find the largest among three numbers.

Theory :

Python if else Statement

Along with the if statement, else keyword can also be optionally used. Python if else statement provides an alternate block of statements to be executed if the Boolean expression (in if statement) is not true.

if else Syntax

The syntax of an if else statement is:

```
if boolean_expression:
 # code block to be executed
 # when boolean_expression is true
else:
  # code block to be executed
  # when boolean_expression is false
```

Algorithm

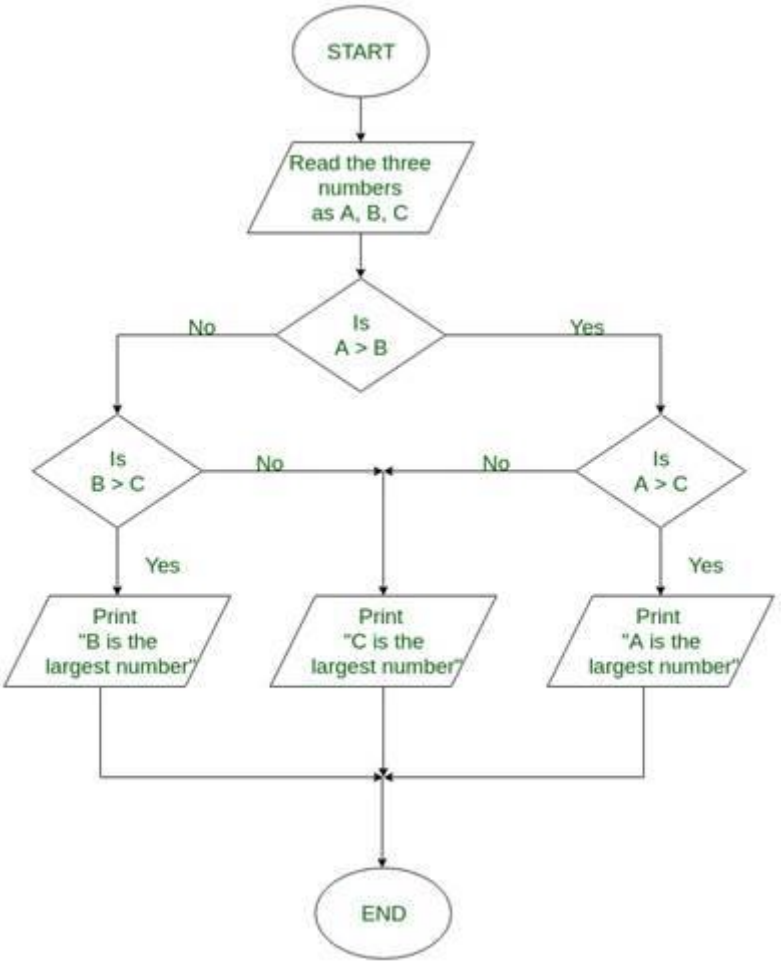
Step 1: input three user input number.

Step2: Add three numbers to the list.

Step 3: Compare numbers with each other or use max() function to find the greatest number max(list).

Step 4: And finally we will print the maximum number.

Flowchart :



Program :

```
# Python program to find the largest number among the three input numbers

# change the values of num1, num2 and num3

# for a different result

num1 = 10

num2 = 14

num3 = 12
```

```
# uncomment following lines to take three numbers from user
#num1 = float(input("Enter first number: "))
#num2 = float(input("Enter second number: "))
#num3 = float(input("Enter third number: "))

if (num1 >= num2) and (num1 >= num3):
   largest = num1
elif (num2 >= num1) and (num2 >= num3):
   largest = num2
else:
   largest = num3
print("The largest number is", largest)
```

Using max() function

```
a = 10
b = 25
c = 15
largest = max(a, b, c)
print("The largest number among", a, ",", b, "and", c, "is", largest)
```

OUTPUT :

```
The largest number among 10, 25, and 15 is 25
```

CONCLUSION :

QUESTIONS :

1. **What is the purpose of the if statement in Python?**

2. **What is the purpose of using nested if-else statements in Python?**

3. **Write the syntax of the nested if else statement.**

4. **Write the definition of max() function.**

EXPERIMENT NO: 12

TITLE: - Study of concept of loops in python

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO. :

Aim : Write a Python program to print all prime numbers up to a number.

Theory :

For Loops in Python

For Loops in Python are a special type of loop statement that is used for sequential traversal. Python For loop is used for iterating over an iterable like a String, Tuple, List, Set, or Dictionary.

In Python, there is no C style for loop, i.e., for (i=0; I <n; i++). The For Loops in Python is similar to each loop in other languages, used for sequential traversals.

Python For Loop Syntax

for var in iterable:

    # statements

Prime numbers are integers greater than 1 that have no positive divisors other than 1 and themselves. To find all prime numbers up to a given number, we can use a simple algorithm that iterates through each number in the range and checks if it is prime.

Algorithm:

1. Start with a list to store the prime numbers.
2. Iterate over each number num in the range from 2 to the given number.
3. For each num, check if it is divisible by any number from 2 to the square root of num. If it is divisible, it is not prime. Otherwise, it is prime and should be added to the list.
4. Finally, print the list of prime numbers.

Program :

```
print("The Prime Numbers up to the limit 50 are : ")

for num in range (0,51):
        for j in range(2, num):

    If num % j == 0 :

        break

  else:
        print(num)
```

OUTPUT :

CONCLUSION :

QUESTIONS :

1. **What is the purpose of the for loop in Python?**

2. **Explain the difference between the range() function and a for loop in Python.**

EXPERIMENT NO: 13

TITLE: -Study of loop in Python

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO :

AIM: Write a Python program to accept a number from user and print digits of number in a reverse order.

PROBLEM DEFINITION : Write a Python program to accept a number from user and print digits of number in a reverse order.

THEORY:

While Loop

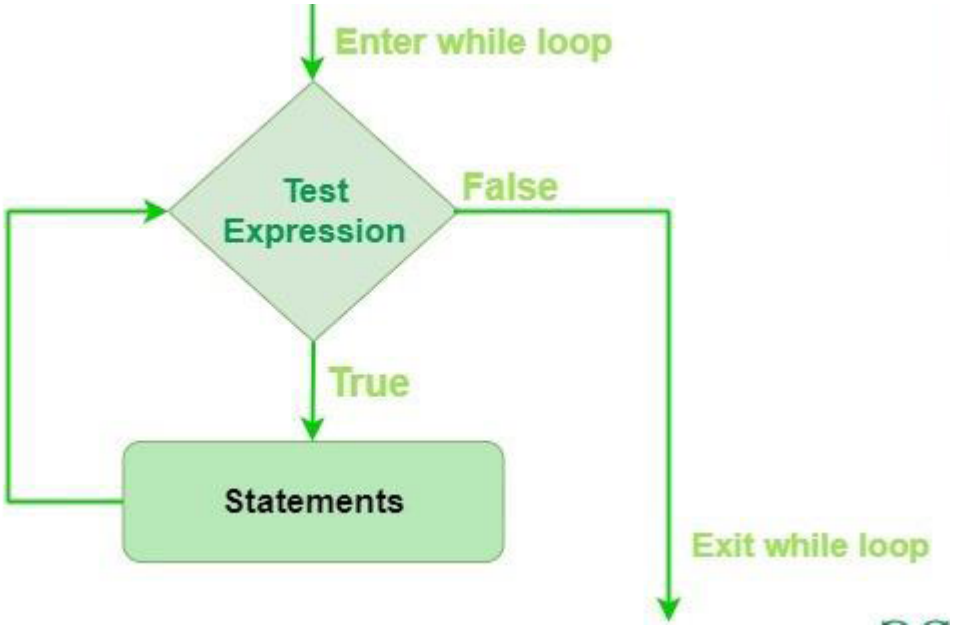
Python While Loop is used to execute a block of statements repeatedly until a given condition is satisfied. When the condition becomes false, the line immediately after the loop in the program is executed.

Syntax of while loop in Python

while expression:

statement(s)

Flowchart :



PROGRAM CODE:

```
num = int(input("Enter a number: "))

reverse = 0

while num > 0:

rem = num % 10 # extract the last digit

reverse = reverse * 10 + rem # append rem to the end of the reversed number

num //= 10 # drop the last digit

print("Reversed number: ", reverse)
```

OUTPUT :

CONCLUSION :

QUESTIONS :

1. What is a while loop in Python?

2. How does a while loop differ from a for loop?

3. How do you create an infinite loop using while?

EXPERIMENT NO: 14

TITLE: - Study of String in Python

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO :

AIM: Study of String in Python

PROBLEM DEFINITION : Write a python program that accepts a string from user and perform following string
operations- i. Calculate length of string ii. String reversal iii. Equality check of two strings.

.

THEORY:

Python String

A String is a data structure in Python Programming that represents a sequence of characters. It is an immutable data type, meaning that once you have created a string, you cannot change it.

Python String are used widely in many different applications, such as storing and manipulating text data, representing names, addresses, and other types of data that can be represented as text.

Syntax of String Data Type in Python

string_variable = 'Hello, world!'

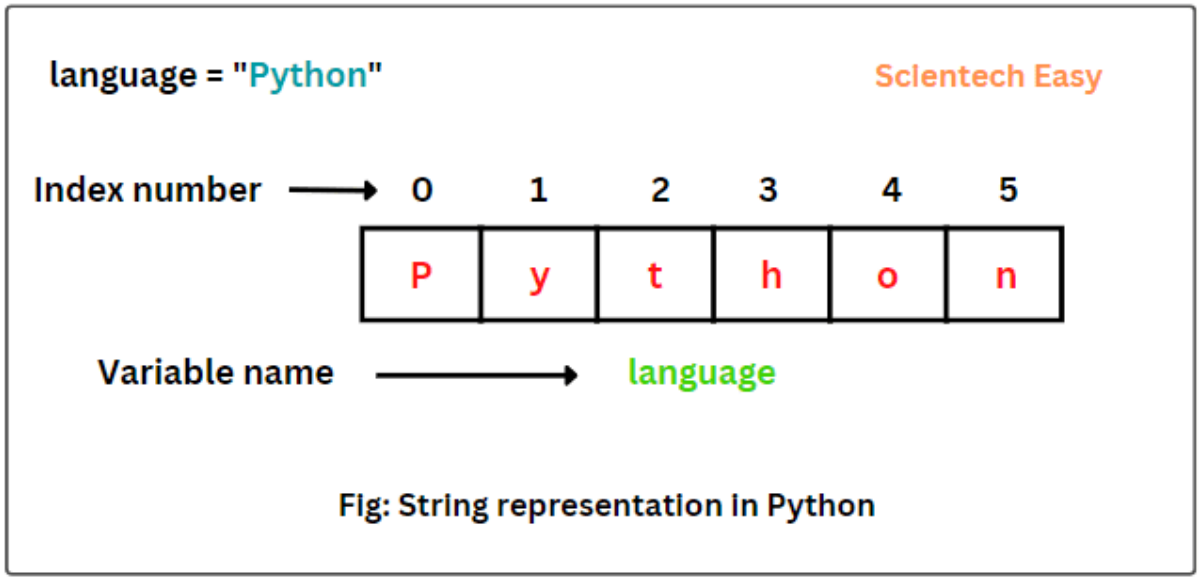


Fig: String representation in Python

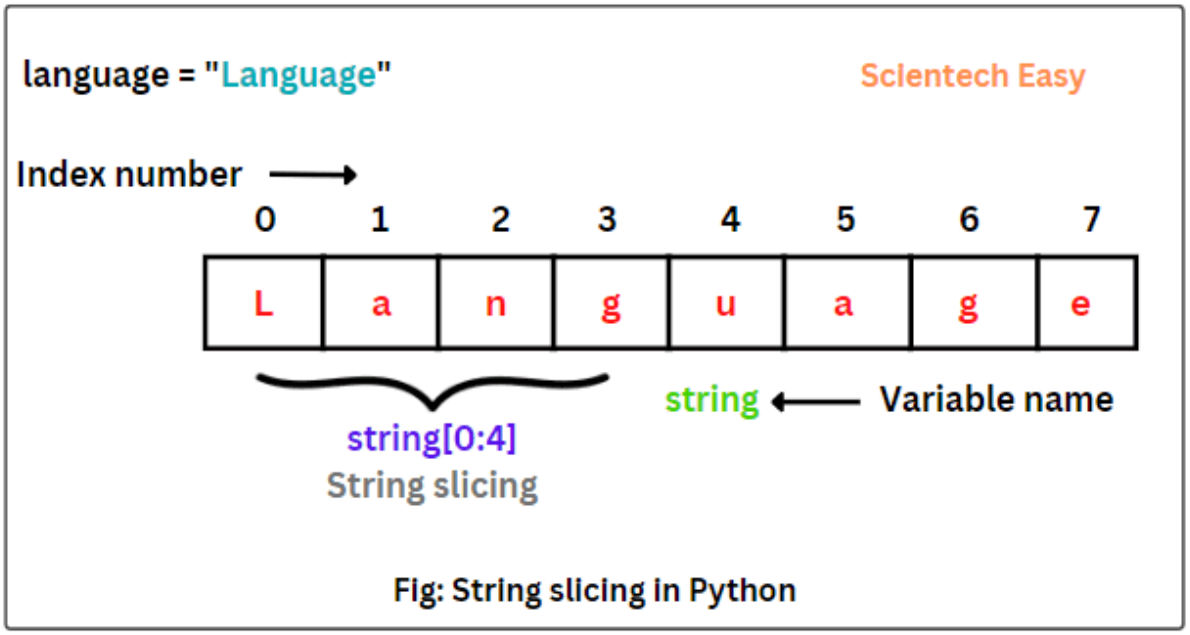


Fig: String slicing in Python

PROGRAM :

```
def string_function():
# Accepting a string from the user
user_string = input("Enter a string: ")
# i. Calculate the length of the string
string_length = len(user_string)
print(f"Length of the string: {string_length}")
# ii. String reversal
reversed_string = user_string[::-1]
print(f"Reversed string: {reversed_string}")
# iii. Equality check of two strings
another_string = input("Enter another string to compare: ")
if user_string == another_string:
print("The strings are equal.")
else:
print("The strings are not equal.")
string_function()
```

OUTPUT :

CONCLUSION:

QUESTIONS :

1. What is a string in Python?

2. How do you create a string in Python?

3. How can you access individual characters in a string?

EXPERIMENT NO: 15

TITLE: - Study of class,object and constructor in python.

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO:

AIM: Create class EMPLOYEE for storing details (Name, Designation, gender, Date of Joining and Salary).Define function members to compute a)total number of employees in an organization b) count of male and female employee c) Employee with salary more than 10,000d) Employee with designation "Asst Manager".

PROBLEM DEFINITION : Study of class,object and constructor in python**.**

THEORY: A class in Python is a blueprint for creating objects. We define an EMPLOYEE class to store and manage employee details. This class will have attributes for each piece of employee information and methods to perform various operations

PROGRAM CODE:

```
class EMPLOYEE:
# Class variable to keep track of total employees and gender count
total_employees = 0
male_count = 0
female_count = 0
high_salary_count = 0
asst_manager_count = 0
def __init__(self, name, designation, gender, date_of_joining, salary):
self.name = name
self.designation = designation
self.gender = gender
self.date_of_joining = date_of_joining
self.salary = salary
EMPLOYEE.total_employees += 1
if gender.lower() == 'male':
EMPLOYEE.male_count += 1
```

```
elif gender.lower() == 'female':
EMPLOYEE.female_count += 1
if salary > 10000:
EMPLOYEE.high_salary_count += 1
if designation.lower() == "asst manager":
EMPLOYEE.asst_manager_count += 1
@classmethod
def get_total_employees(self):
return self.total_employees
@classmethod
def get_gender_count(cls):
return {"male": cls.male_count, "female": cls.female_count}
@classmethod
def get_high_salary_count(cls):
return cls.high_salary_count
@classmethod
def get_asst_manager_count(cls):
return cls.asst_manager_count
e1 = EMPLOYEE("John Doe", "Manager", "Male", "2021-01-15", 12000)
e2 = EMPLOYEE("Jane Smith", "Asst Manager", "Female", "2020-06-10", 9500)
e3 = EMPLOYEE("Alice Johnson", "Developer", "Female", "2019-03-25", 10500)
e4 = EMPLOYEE("Bob Brown", "Asst Manager", "Male", "2018-07-30", 11000)
print("Total number of employees:", EMPLOYEE.get_total_employees())
print("Gender count:", EMPLOYEE.get_gender_count())
print("Number of employees with salary more than 10,000:", EMPLOYEE.get_high_salary_count())
print("Number of Asst Managers:", EMPLOYEE.get_asst_manager_count())
```

OUTPUT :

CONCLUSION:

QUESTIONS :

1. What is a class in Python?

2. What is an object in Python?

3. How do you create a class in Python?

EXPERIMENT NO: 16

TITLE: - Study of operator overloading in python.

CLASS: - First Year (SEM -II) **BATCH: -**

DATE:

ROLL NO:

AIM: Write a program in python to perform following operations on complex numbers.
1)Add 2) Subtract 3) Multiply 4) Divide
Use operator overloading for these operations.

PROBLEM DEFINITION : Study of operator overloading in python

THEORY:
Operator Overloading

Operator overloading allows us to define how operators behave with objects of user-defined classes. By overloading operators, we can use standard arithmetic operators (+, -, *, /) to perform operations on complex numbers in a natural and intuitive way. This makes the code more readable and easier to work with.

Complex numbers are numbers that have both a real and an imaginary part, typically represented in the form $a+b\ i$a+bi, where a is the real part and b is the imaginary part. In Python, we can define our own class to handle complex numbers and use operator overloading to perform arithmetic operations such as addition, subtraction, multiplication, and division.

PROGRAM :

```
class ComplexNumber:
def __init__(self, real, imag):
        self.real = real
        self.imag = imag
        def __str__(self):
        return f"{self.real} + {self.imag}i"
```

```
def __add__(self, other):
return ComplexNumber(self.real + other.real, self.imag + other.imag)
def __sub__(self, other):
return ComplexNumber(self.real - other.real, self.imag - other.imag)
def __mul__(self, other):
real_part = self.real * other.real - self.imag * other.imag
imag_part = self.real * other.imag + self.imag * other.real
return ComplexNumber(real_part, imag_part)
def __truediv__(self, other):
denominator = other.real**2 + other.imag**2
real_part = (self.real * other.real + self.imag * other.imag) / denominator
imag_part = (self.imag * other.real - self.real * other.imag) / denominator
return ComplexNumber(real_part, imag_part)
# Example usage
c1 = ComplexNumber(3, 2)
c2 = ComplexNumber(1, 7)
print("Addition:", c1 + c2)
print("Subtraction:", c1 - c2)
print("Multiplication:", c1 * c2)
print("Division:", c1 / c2)
```

OUTPUT :

CONCLUSION:

QUESTIONS :

1. What is a complex number?

2. How do you represent a complex number in Python using a class?

3. What is operator overloading?

4. What method do you use to overload the subtraction operator in the class?